WHAT IS CLAIMED IS:

1. In a pipelined processor, a method for reducing pipeline stalls caused by branching, said method comprising the steps of:

prefetching instructions into a first stage of said pipeline;

propagating instructions into one or more subsequent stages of said pipeline;

computing a conditional outcome in one of said subsequent stages;

concurrently with processing at a specified stage in said pipeline, analyzing one or more instruction op-codes to determine whether a cacheable branch instruction is present, and, if said branch instruction is present, sending a tag relating to said branch instruction to a branch cache;

determining, in response to said conditional outcome, whether a branch is to be taken, and, if said branch is to be taken, sending a branch taken signal to said branch cache;

if the conditional outcome indicates a branch is not to be taken, continuing to fetch instructions into said pipeline and to execute said instructions; and

on receipt of said current branch tag, said branch cache performing the steps of:

examining a collection of stored branch tags to find a stored branch tag which matches said current branch tag;

if said current branch tag is not found in said collection of stored branch tags and said branch is to be taken:

signaling a cache miss;

causing said pipeline to fill one or more designated pipeline stages starting at a branch target address, said designated pipeline stages being pipeline stages that stall according to said branch, said branch cache storing said current branch tag and one or more instructions contained within said designated pipeline stages;

and

if said branch taken signal is received and said current branch tag is found in said collection of stored branch tags:

signaling a cache hit;

sending a branch target address to the prefetch unit so that instruction fetching can proceed from said branch target address; and

providing data stored in said cache to one or more of said designated pipeline stages so that execution can continue without delay irrespective of said conditional outcome.

2. A computer processor comprising:

an instruction pipeline comprising a plurality of stages, each stage containing pipeline data;

a branch cache comprising a plurality of cache lines, each cache line comprising a stored branch tag and stored cache data; and

a branch cache controller configured to:

detect a cacheable branch instruction in one of said pipeline stages;

receive a current branch tag from one of said pipeline stages;

receive conditional information indicative of whether the branch shall be taken;

attempt to match said current branch tag to a stored branch tag for a first cache line;

if said branch is to be taken, signal a cache miss when said attempt to match fails;

if said branch is to be taken, signal a cache hit when said attempt to match succeeds;

in response to said cache miss, store said current branch tag in said stored branch tag of a designated cache line and store in said stored cache data of said designated cache line data from one or more of said pipeline stages which stall in response to said cacheable branch instruction; and

in response to said cache hit, load one or more of said pipeline stages from said stored cache data to avoid a pipeline stall from said cacheable branch instruction.

3. A computer processor comprising:

an instruction pipeline comprising a plurality of stages, each stage containing data;

means for storing data from one or more of said pipeline stages and restoring data to one or more of said pipeline stages; and

means for controlling said means for storing, said means for controlling causing said branch cache to store data from one or more of said pipeline stages in response to execution of a cacheable branch instruction which triggers a cache miss, and causing said means for storing to restore data to one or more of said pipeline stages in response to a cache hit, thereby avoiding pipeline stalls when a cache hit occurs.

4. In a pipelined microsystem such as a microprocessor, DSP, media processor, or microcontroller, a method to load branch instruction information into a branch cache so as to allow the branch instruction to execute subsequently with a reduced or eliminated time penalty by minimizing the amount of information to be cached, the method comprising the steps of:

monitoring the instruction stream in a dispatch unit in a pipeline stage to detect whether a branch instruction of a selected type is present;

when said branch instruction is detected:

signaling to a branch cache control unit that the instruction is present;

making available at least a portion of an address of said branch instruction to said branch cache control unit;

comparing said at least a portion of said address of said branch instruction to a set of cache tags containing branch instruction address related information;

when said branch instruction does not match any tag, filling the branch cache entry so that when said branch instruction is next encountered, the tag will match and the branch target stream can proceed without delay; and

when program execution makes a branch target fetch packet available to be cached to allow the target instruction stream to execute to a target prefetch buffer, performing the steps of:

loading data from said target prefetch buffer into a position in the branch cache line associated with said branch instruction;

setting a counter to a prespecified number, d, corresponding to the maximum possible number of fetch packets that may need to be cached;

decrementing the counter on each subsequent cycle,

loading subsequent fetch packets from the target instruction stream into the branch cache line only when they are fetched; and

exiting the branch cache fill operation when the counter has decremented to a specified number such that the branch cache line is filled with the appropriate number of target prefetch packets that are fetched in the first d time slots when the target instruction stream is executed.

5. The method according to Claim 4, further including the step of loading stall override bits into the branch cache line, said stall override bits indicating for each of the d cycles whether or not the branch cache will supply the target fetch packet during a given cycle.

6. The method according to Claim 4, further including the step of storing a condition field to indicate a register or an execute stage which supplies the conditional branch information so that the branch cache can resolve the branch early.

7. The method according to Claim 4, further including the step of supplying an auxiliary link field which points to a next prefetch buffer of the cache line, said auxiliary link field creating a linked list in a variable-length cache line structure.

8. The method according to Claim 4, further including the step of caching shadow dispatch unit pre-evaluation data to allow a shadow dispatch unit to dispatch instructions using less hardware than said dispatch unit.

9. In a pipelined microsystem such as a microprocessor, DSP, media processor, or microcontroller, a method to service branch cache hits so as to reduce or eliminate cycle loss due to branching, said method comprising the steps of:

monitoring the instruction stream in a pipeline stage to detect whether a branch instruction of a selected type is present;

when said branch instruction is detected:

signaling to a branch cache control unit that the instruction is present; and

making available at least a portion of an address of said branch instruction to the branch cache control unit;

comparing said at least a portion of said address of said branch instruction to a set of tags containing branch instruction address related information;

when said branch instruction does match a tag and said branch is evaluated to be taken, performing the steps of:

reading a target prefetch buffer out of the branch cache and supplying the target prefetch buffer to a shadow dispatch unit;

dispatching said prefetch buffer from said shadow dispatch unit to a multiple execution pipeline in units of execute packets;

prefetching instructions at a full prefetch rate, irrespective of whether multiple cycles are required to dispatch a fetch packet, said prefetching instructions at a full prefetch rated continuing until early pipeline stages catch up to later pipeline stages, whereby the

target instruction stream proceeds at full speed and only a minimum number of fetch packets needed to support full speed execution are fetched from the branch cache.

10. In a pipelined microsystem such as a microprocessor, DSP, media processor, or microcontroller, a method for servicing branch cache hits so as to reduce or eliminate cycle loss due to branching, said method comprising the steps of:

monitoring the instruction stream in a pipeline stage to detect whether a branch instruction of a selected type is present;

when said branch instruction of a selected type is detected:

signaling to a branch cache control unit that the instruction is present; and

making available at least a portion of said branch instruction's address to the branch cache control unit;

comparing said at least a portion of an address of said branch instruction to a set of tags containing branch instruction address related information;

when said branch instruction does match a tag and said branch is evaluated to be taken, performing the steps of:

reading the target prefetch buffer out of the branch cache;

supplying the contents of the target prefetch buffer to a multiplexer which routes the contents of the target prefetch buffer back to the dispatch unit;

dispatching the contents of the target prefetch buffer to said pipeline in units of execute packets;

prefetching instructions by said pipeline at full speed, irrespective of whether it takes multiple cycles to dispatch a fetch packet, until the early pipeline stages catch up to the later pipeline stages, whereby the target instruction stream proceeds at nearly full speed, and only a minimum number of fetch packets needed to support full speed execution are fetched from the branch cache.

11. In a VLIW processor which fetches groups of instructions in fetch packets and dispatches subsets thereof as execute packets in one or more clock cycles, a method for reducing the size of a branch cache which buffers branch target information, the method comprising the steps of:

caching the target prefetch buffer when a branch cache miss is detected; and

caching a variable number of immediately following prefetch buffers, the number of cached prefetched buffers being the number of prefetch buffers that are fetched in the target instruction stream during the first d cycles of execution, where the number d is related to the number of pipeline stages that would otherwise stall when a branch occurs.

12. A branch cache to be used in a multi-issue processor having an address generate portion in a prefetch unit, wherein said processor dispatches in each clock cycle variable numbers of instructions contained in each fetch packet, said cache comprising:

a plurality of lines, each line comprising:

a tag field which holds information relating to the addresses of branch instructions, said information including address information of branch instructions of a selected type or types;

a branch address field which holds an address near to the branch target address, so that this near address can be forwarded to the program address generate portion of the prefetch unit for target instruction stream fetching;

a prefetch buffer field which holds the first prefetch buffer of the target instruction stream;

at least one link field which indicates whether more prefetch buffers are associated with said tag field; and

at least one extra prefetch buffer field.

13. The branch cache as defined in Claim 12, wherein a number of said least one extra prefetch buffer field is determined by initial prefetch activity of the target instruction stream.

14. The branch cache as defined in Claim 12, wherein each cache line additionally comprises a pipeline stall override field which signals the prefetch unit to continue to fetch instructions when there would otherwise be a pipeline stall due to multiple execute packets being dispatched from a single target fetch packet.

15. The branch cache as defined in Claim 12, wherein additional prefetch buffers of the cache line are arranged in a linked list structure.

16. A method to fill an instruction pipeline after a branch instruction is detected which selects a target instruction stream, the method comprising the steps of:

reading a prefetch buffer out of the branch cache line associated with the instruction which caused the branch cache hit;

sending the cached prefetch buffer to a shadow dispatch unit;

routing the output of the shadow dispatch unit to a multiplexer which selects instruction information from a dispatch unit in the execution pipeline or from a shadow dispatch unit;

providing a select signal which forces the multiplexer to select the cached fetch packet from the shadow dispatch unit;

forwarding the fetch packet to decoder stages of an execution pipeline in units of execute packets;

allowing the prefetch stages of the instruction pipeline to continue functioning irrespective of how many execute packets are in each fetch packet until the instruction pipeline is filled; and

supplying the requisite number of fetch packets from the branch cache to allow the target instruction stream to proceed without adding extra delay cycles.

17. A method to fill an instruction pipeline after a branch instruction is detected which selects a target instruction stream, the method comprising steps of:

reading a prefetch buffer out of the branch cache line associated with the instruction which caused the branch cache hit;

sending the cached prefetch buffer to a dispatch unit;

routing the output of the shadow dispatch unit to decoder stages of an execution pipeline in units of execute packets;

allowing the prefetch stages of the instruction pipeline to continue functioning irrespective of how many execute packets are in each fetch packet until the instruction pipeline is filled; and

supplying the requisite number of fetch packets from the branch cache to allow the target instruction stream to proceed without adding extra delay cycles.

18. A method to detect and control the branch cache related processing of branch instructions in processing systems comprising a first cacheable branch instruction type and a second non-cacheable branch instruction type, the method comprising the steps of:

evaluating bits located in an instruction that passes through a selected stage of an instruction pipeline to determine whether said instruction corresponds to a cacheable branch instruction;

if said instruction corresponds to a cacheable branch instruction, evaluating a condition and a tag associated with said instruction to determine whether data needs to be read out of a branch target buffer; and

if said instruction is not a branch instruction or is a non-cacheable branch instruction, continuing processing of said instruction and aborting any subsequent branch cache processing for said instruction.